FlavorShare

Recipe Sharing Platform

Team Members

Name	Role
Corneliu Floarea	Developer

Live Demo:
https://recipe-platform-qior.onrender.com

1. Introduction

FlavorShare is a full-stack recipe sharing platform that allows users to discover, create, and share recipes with a community of food enthusiasts. The application demonstrates comprehensive web development implmentations including user authentication, database management, file uploads, and responsive design.

Project Objectives

- Implement secure user authentication with session management
- Create a complete CRUD system for recipe management
- Enable image uploads with cloud storage
- Build a review and rating system for community engagement
- Develop a responsive UI that works across all devices
- Deploy to production with Render.com hosting

Key Features

- User registration and login with encrypted passwords
- Browse, search, and filter recipes by category, difficulty, and dietary restrictions
- Create recipes with ingredients, instructions, and images
- Write reviews and rate recipes
- Save favorite recipes to your profile
- Persistent image storage

2. Design Considerations

Technology Stack

Layer	Technologies
Frontend	HTML5, CSS, JavaScript
Backend	Node.js, Express.js
Database	MongoDB Atlas
Image Storage	Cloudinary CDN
Hosting	Render.com
Authentication	Express-session with MongoDB store

Database Design

The application uses MongoDB with three main collections:

- **Users:** Stores user accounts with bcrypt-hashed passwords, profile info, and favorites array
- **Recipes:** Contains recipe data including ingredients, instructions, images (Cloudinary URLs), ratings, and author references
- **Reviews:** Stores user reviews with ratings, linked to both users and recipes

Security Implementation

- **Password Hashing:** bcryptjs with salt rounds for secure password storage
- **Session Management:** express-session with MongoDB store
- **Input Validation:** express-validator for server-side validation, Mongoose schema validation
- **File Upload Security:** Multer with file type filtering, size limits (5MB max)

3. User Interface Design

The UI follows a mobile-first responsive design approach with a custom CSS framework. Key design principles include:

- **Simplicity:** Clean, uncluttered layouts with clear visual hierarchy
- **Consistency:** Reusable components (cards, buttons, forms) with CSS variables
- **Accessibility:** Semantic HTML, proper contrast ratios, keyboard navigation
- **Responsiveness:** Breakpoints at 480px, 768px, and 1024px for all device sizes

- No prototypes were made. Designed UI as I go. Also took inspiration from previous web project (like navbar, routing, authentication, etc).

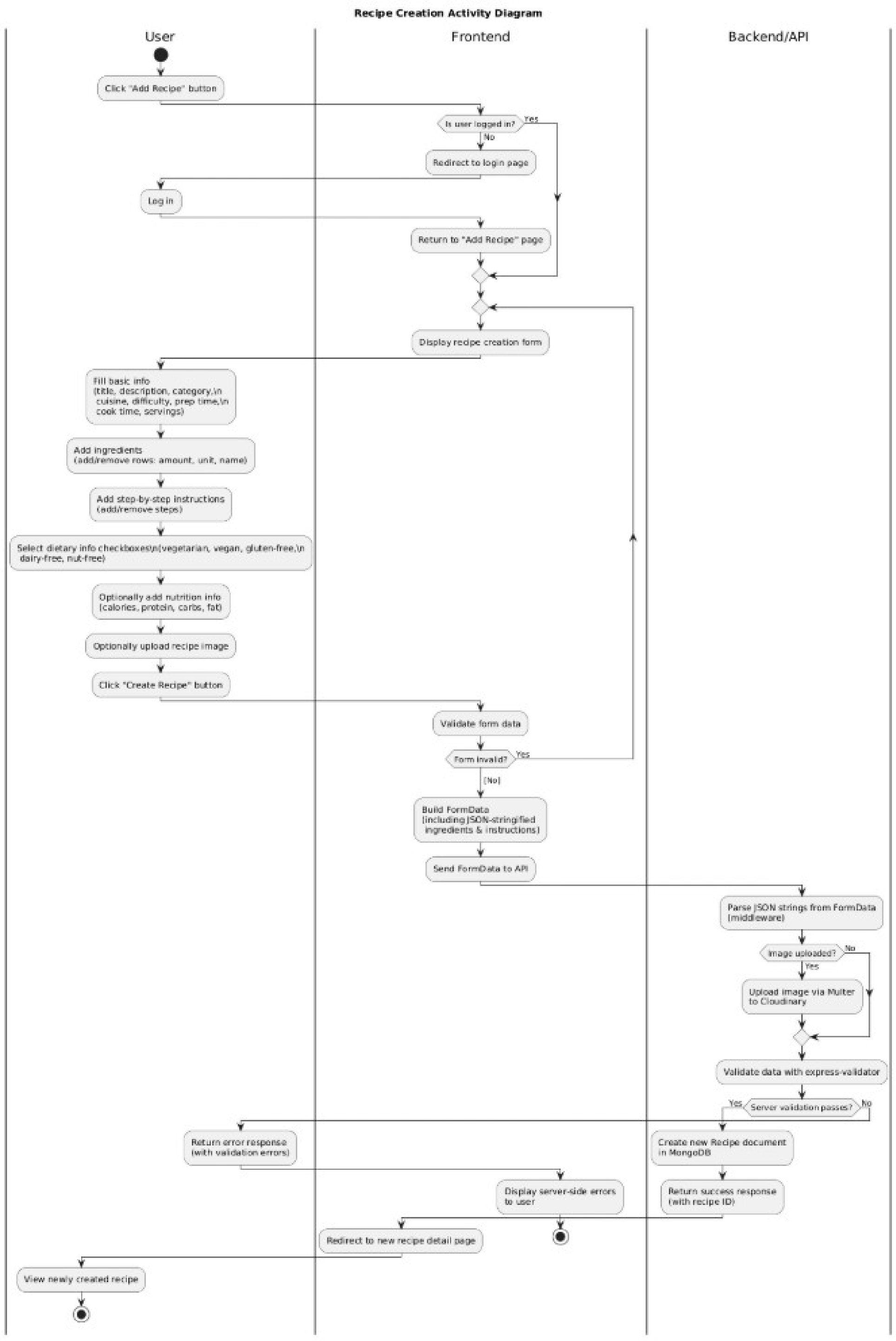
4. System Components

Component	Description
server.js	Main Express server with middleware, session config, trust proxy
routes/auth.js	Authentication routes: register, login, logout, profile
routes/recipes.js	Recipe CRUD, search, filtering, favorites, Cloudinary integration
config/cloudinary.js	Cloudinary configuration for persistent image storage
middleware/	Auth checks, validation, FormData parsing for JSON arrays
models/	Mongoose schemas for User, Recipe, Review with validation

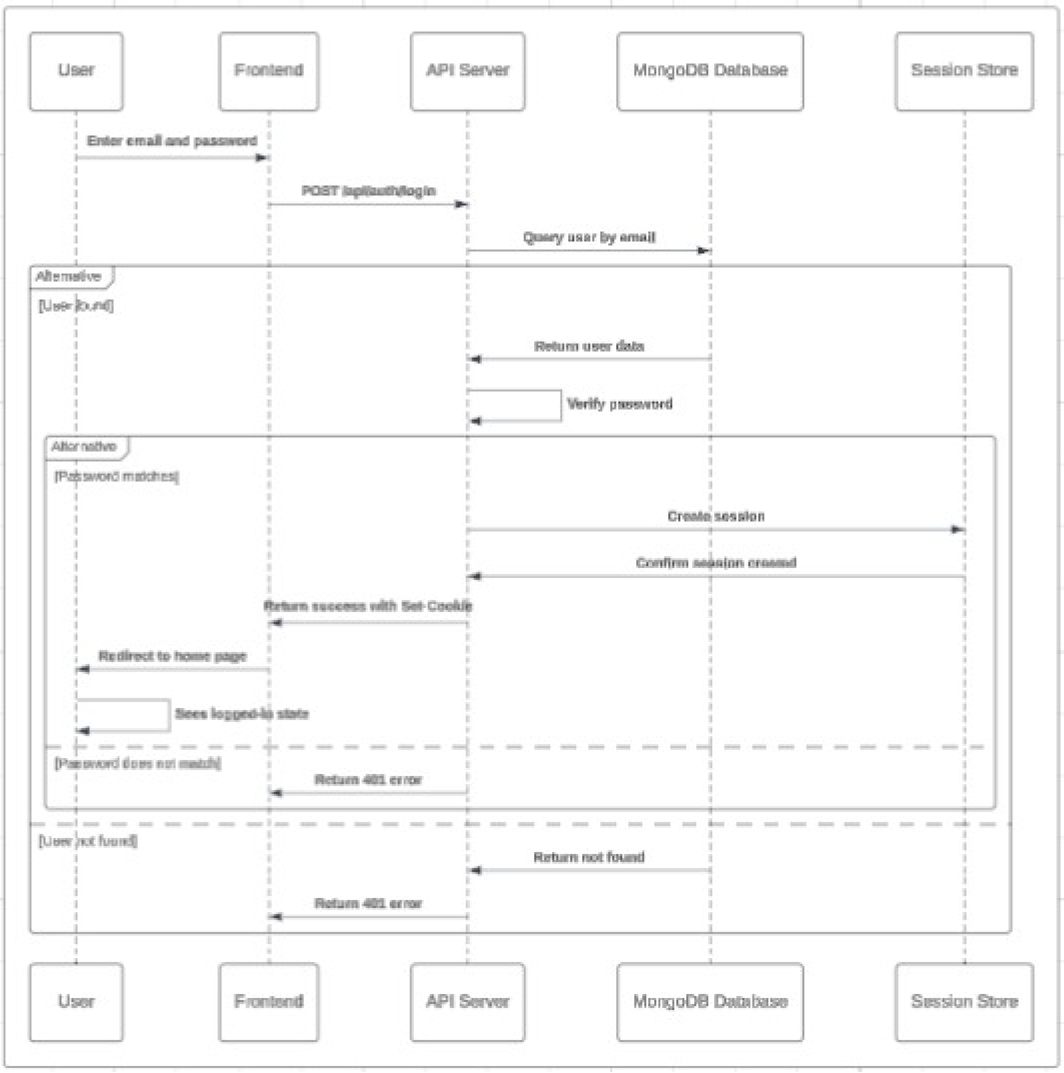
5. UML Diagrams

Activity Diagram: Recipe Creation Flow

This diagram shows the complete flow when a user creates a new recipe:


Recipe Creation Activity Diagram
User
Frontend
Backend/API
Click "Add Recipe" button
Is user logged in?
Yes
No
Redirect to login page
Log in
Return to "Add Recipe" page
Display recipe creation form
Fill basic info (title, description, category,\n cuisine, difficulty, prep time,\n cook time, servings)
Add ingredients (add/remove rows: amount, unit, name)
Add step-by-step instructions (add/remove steps)
Select dietary info checkboxes\n(vegetarian, vegan, gluten-free,\n dairy-free, nut-free)
Optionally add nutrition info (calories, protein, carbs, fat)
Optionally upload recipe image
Click "Create Recipe" button
Validate form data
Form invalid?
Yes
[No]
Build FormData (including JSON-stringified ingredients & instructions)
Send FormData to API
Parse JSON strings from FormData (middleware)
Image uploaded?
No
Yes
Upload image via Multer to Cloudinary
Validate data with express-validator
Server validation passes?
Yes
No
Return error response (with validation errors)
Create new Recipe document in MongoDB
Display server-side errors to user
Return success response (with recipe ID)
Redirect to new recipe detail page
View newly created recipe

Sequence Diagram: User Login



6. Summary & Lessons Learned

Challenges Faced

- **Session Management in Production:** Cookies weren't being set on Render due to proxy settings. Required adding 'trust proxy' and 'sameSite: none' configuration.
- **Image Persistence:** Render's ephemeral filesystem deleted uploaded images on server restart. Solved by integrating Cloudinary for cloud-based image storage.
- **FormData Parsing:** Arrays sent via FormData arrived as strings. Created parseFormData middleware to convert JSON strings before validation.
- **Double Form Submission:** Recipe form was submitting twice due to duplicate event listeners. Fixed by removing redundant DOMContentLoaded handler.
- **Package Compatibility:** multer-storage-cloudinary v2 vs v4 had different APIs. Had to match the import syntax to the installed version.

Key Lessons

- Always test in production-like environments early to catch deployment-specific issues
- Cloud services (Cloudinary, MongoDB Atlas) simplify deployment and provide better reliability
- Browser DevTools Network tab is essential for debugging API issues
- Environment variables should be properly configured for both development and production

API Endpoints Reference

Method	Endpoint	Description
POST	/api/auth/register	Create new user account
POST	/api/auth/login	Authenticate user
GET	/api/recipes	Get all recipes (with filters)
POST	/api/recipes	Create new recipe
PUT	/api/recipes/:id	Update recipe
DELETE	/api/recipes/:id	Delete recipe
POST	/api/reviews/recipe/:id	Add review to recipe